Julius-Maximilians-
UNIVERSITÄT WÜRZBURG
Lehrstuhl für
INFORMATIK I
Effiziente Algorithmen und wissensbasierte Systeme
Institut für Informatik

Advanced Algorithms

Winter term 2019/20

Lecture 2. Approaches for Vertex Cover

Steven Chaplick & Alexander Wolff
Chair for Computer Science I

Approaches to NP-Hard Problems

- Exponential-time algorithms, e.g., backtracking
- Approximation algorithms:
  trade off: quality against running time
- Heuristics: experiments on benchmarks
- Randomization: find a needle in the haystack

Example: Vertex Cover

Def. (Recall)
Let $G = (V, E)$ be an undirected graph.
$C \subseteq V$ is a *vertex cover* of G
if, for all $uv \in E$, it holds that $\{u, v\} \cap C \neq \emptyset$.

Prob. *Minimum Vertex Cover* – optimization problem

Given: graph G
Find: smallest (minimum) vertex cover of G

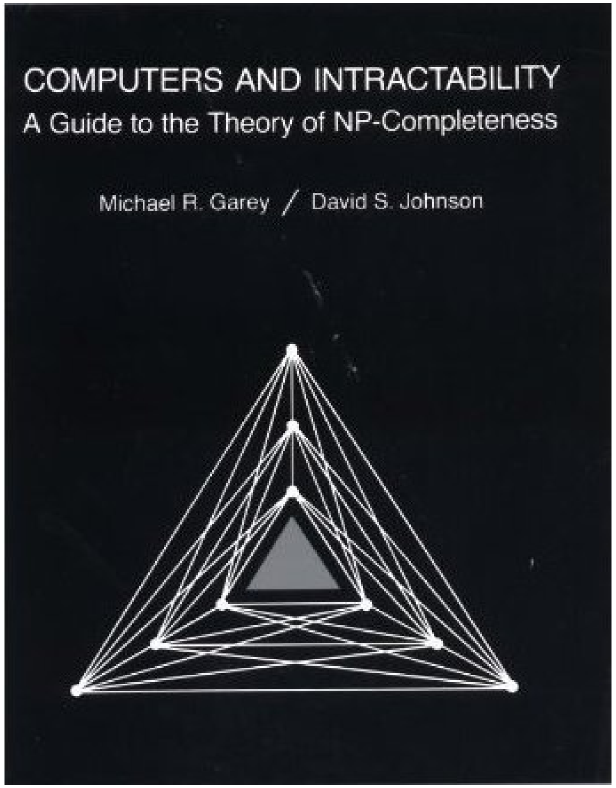
Prob. *k-Vertex Cover (k-VC)* – decision problem

Given: graph G, natural number k
Find: vertex cover of size $\leq k$ of G –
if there is any – otherwise return "no".

Previous Work

- One of the first problems whose NP-hardness has been shown (SAT $\preceq_p$ CLIQUE $\preceq_p$ VC $\preceq_p \ldots$) [Karp, 1972]

- One of the six "basic" NP-hard problems. [Garey & Johnson, 1979]



- Approximable...
  A maximal matching "yields" a 2-approximation.

- ...but not arbitrarily well:
  There is no 1.36-approximation for VC.
  if $\mathcal{P} \neq \mathcal{NP}$.
  [Dinur & Safra, 2004]

Approximation Algorithms

Let Π be a ~~minimization~~ maximization problem and $\cancel{\alpha \in \mathbb{Q}^+}$ $\alpha \colon \mathbb{N} \to \mathbb{Q}$.
A factor-α-approximation algorithm for Π is an efficient algorithm which provides a feasible solution $s \in S_\Pi(I)$ for **any** instance $I \in D_\Pi$ such that:

$$\frac{\mathrm{obj}_\Pi(I, s)}{\mathrm{OPT}_\Pi(I)} \overset{\ge}{\cancel{\le}} \cancel{\alpha}. \quad \alpha(|I|)$$

A little exercise. For Vertex Cover –

- What are the *instances* (D_Π)?
- What are the *feasible solutions* ($S_\Pi(I)$) for an instance I?
- What is the *objective function* $\mathrm{obj}_\Pi(I, s)$ for I and s?
- What is the value $\mathrm{OPT}_\Pi(I)$ of the objective function of an optimal solution for I?

Approximation Alg. for Vertex Cover

Ideas?

- Edge-Greedy
- Vertex-Greedy (see Exercises)
- maximal edge covers

How can we measure the quality of a feasible solution?

Problem: How can we estimate $\frac{\mathrm{obj}_\Pi(I,s)}{\mathrm{OPT}}$ when it is hard to calculate OPT?

Idea: Find a "good" lower bound $U \leq \mathrm{OPT}$ for OPT and compare it to our approximate solution.

$$\frac{\mathrm{obj}_\Pi(I,s)}{\mathrm{OPT}} \leq \frac{\mathrm{obj}_\Pi(I,s)}{U}$$

Lower Bound by Matchings

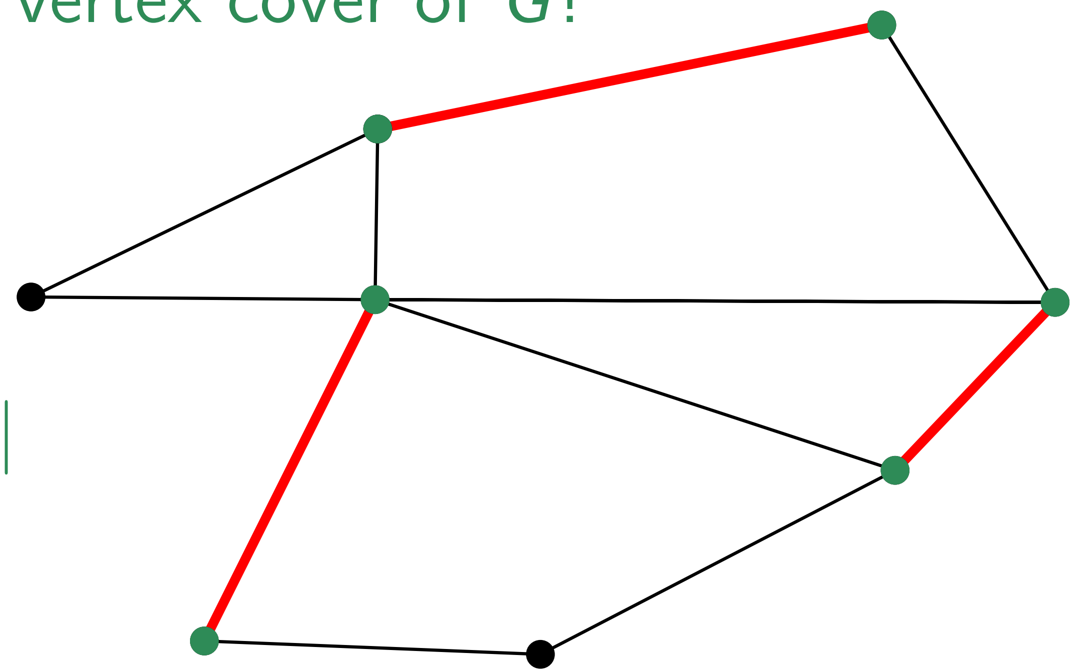
An edge set $M \subseteq E$ of a graph $G = (V, E)$ is a **matching** when no vertex of G is incident to two edges in M .

M is **maximal** when there is no matching M' with $M' \supsetneq M$.

vertex cover of G!

$\mathrm{OPT} \geq \cancel{3}\ |M|$

$\mathrm{OPT} \leq 2 \cdot |M|$



Approximation Alg. for Vertex Cover

Combinatorial Algorithm for Vertex Cover (G)

$M \leftarrow \emptyset$
foreach $e \in E(G)$ **do**
  if e is not adjacent to an edge in M **then**
    $M \leftarrow M \cup \{e\}$
return $\{\, u, v \mid uv \in M \,\}$

Theorem. The above algorithm is a factor-2 approximation algorithm for Vertex Cover.

Proof. 1. Feasibility.
2. Quality of the solutions.

Try!

A Tool for Designing Approximation Algo's

- Formulate the problem as an integer linear program (ILP).
- Relax the ILP, that is, replace integrality by non-negativity.
- Round an optimal solution of the relaxation to an approximate integral solution.

Let's do this for Vertex Cover:

- Let G be the given graph.

$$\begin{array}{lll} \text{Minimize} & \sum_{v \in V} x_v & \\ \text{subject to} & x_u + x_v \geq 1 & \text{for each } uv \in E(G) \\ & \cancel{x_v \in \{0,1\}} & \text{for each } v \in V(G) \end{array}$$

- Relax the ILP: $x_v \geq 0$
- Round the LP solution: Return $\left\{v \in V(G) \colon x_v \geq \frac{1}{2}\right\}$.

LP-Rounding for Vertex Cover

Theorem. The LP-rounding algorithm is a factor-2 approximation algorithm for Vertex Cover.

Proof. 1. Feasibility.
2. Quality of the solutions.

Try!

Question. What is the lower bound that we used here?

The LP solution! $\sum_v x_v \le \text{OPT} \le \sum x'_v \le \sum 2x_v$.

(rounded solution)

How do the 2 algorithms compare? What are their pros & cons?

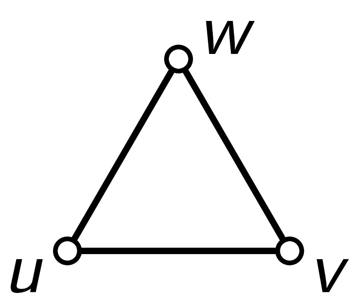
– The combinatorial algorithm is fast and easy to implement.

– The LP-rounding algorithm solves a more general problem:
Minimize $\sum_{v \in V} c_v x_v$ – Weighted Vertex Cover!

Integrality Gap

By how much can LP and ILP solutions differ?

Example:



The LP solution is $\mathrm{OPT}^\star = 1.5$

$x \equiv 0.5$

The ILP solution is $\mathrm{OPT} = 2.0$

$e.g., x = (1, 1, 0)$

Def. *Integrality gap* $= \sup_{I \in D_\Pi} \dfrac{\mathrm{OPT}(I)}{\mathrm{OPT}^\star(I)} \geq \dfrac{4}{3}$ for vertex cover.

Exercise: What is the largest integrality gap for Vertex Cover you can find?

Exercise:
- Draw the three constraints in the VC ILP for C_3 into a 3D coordinate systems.
- Add all optimal LP/ILP solutions.

Polyhedral Insights

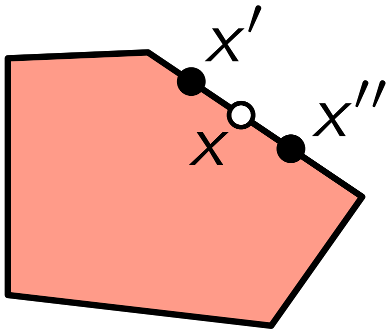
Let $V^- = \{v : 0 < x_v < 1/2\}$ and $V^+ = \{v : 1/2 < x_v < 1\}$.

$\varepsilon := \min\left\{ \min_{v \in V^-}\{x_v, 1/2 - x_v\}, \min_{v \in V^+}\{1 - x_v, x_v - 1/2\}\right\}$

Let x be an (optimal) LP solution.

Let $x'_v = \begin{cases} x_v - \varepsilon & \text{if } v \in V^- \\ x_v + \varepsilon & \text{if } v \in V^+ \\ x_v & \text{else.} \end{cases}$ Let $x''_v = \begin{cases} x_v - \varepsilon & \text{if } v \in V^+ \\ x_v + \varepsilon & \text{if } v \in V^- \\ x_v & \text{else.} \end{cases}$

Then x' and x'' are feasible LP solutions and $x = (x' + x'')/2$.
What does this mean geometrically?



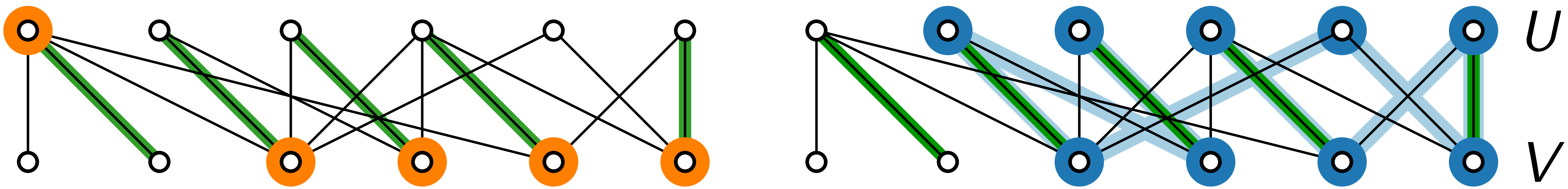
Note: x can't be a corner of the polyhedron if

$V^- \neq \emptyset \quad (\Leftrightarrow V^+ \neq \emptyset)$

So if x is an extreme point, $V^- = \emptyset = V^+$.
In other words, the VC polytope is half integral!

König's Theorem

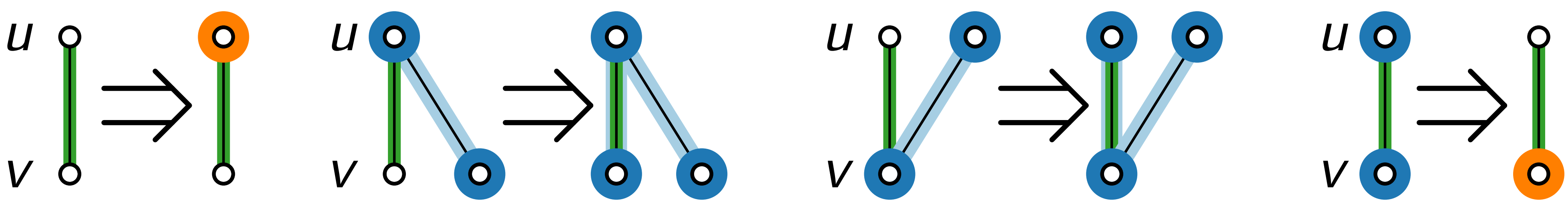
Theorem. Let $G = (U \cup V, E)$ be a bipartite graph, C a minimum vertex cover, and M a maximum matching. Then $|C| = |M|$.



Z: all unmatched vertices in U
+ all vertices that are reachable via alternating paths

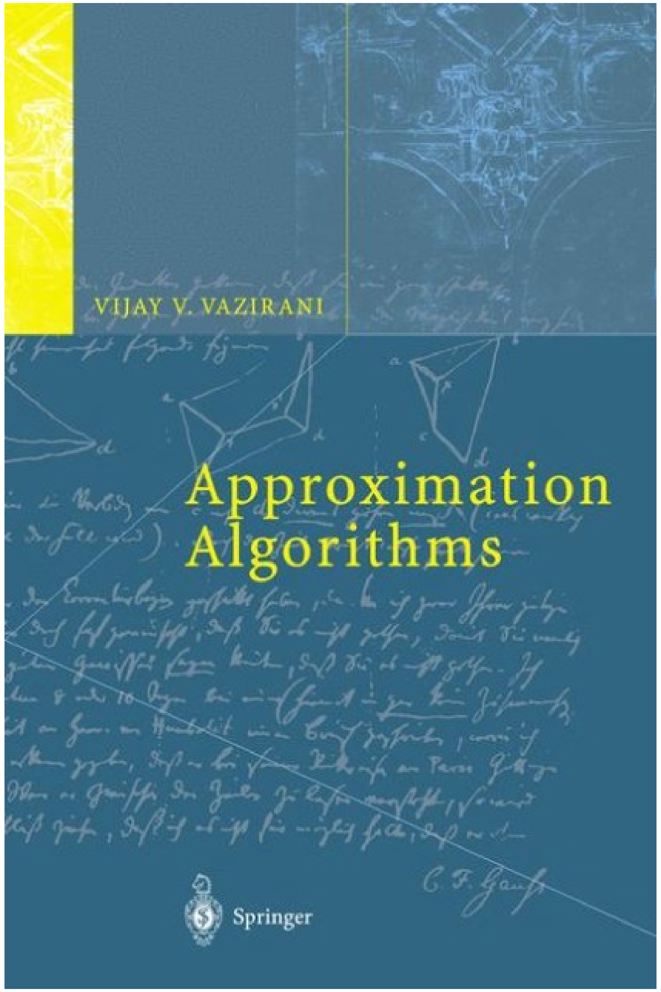
C: $(V \cap Z) \cup (U \setminus Z)$

Lemma. Each edge in M has exactly one vertex in C.

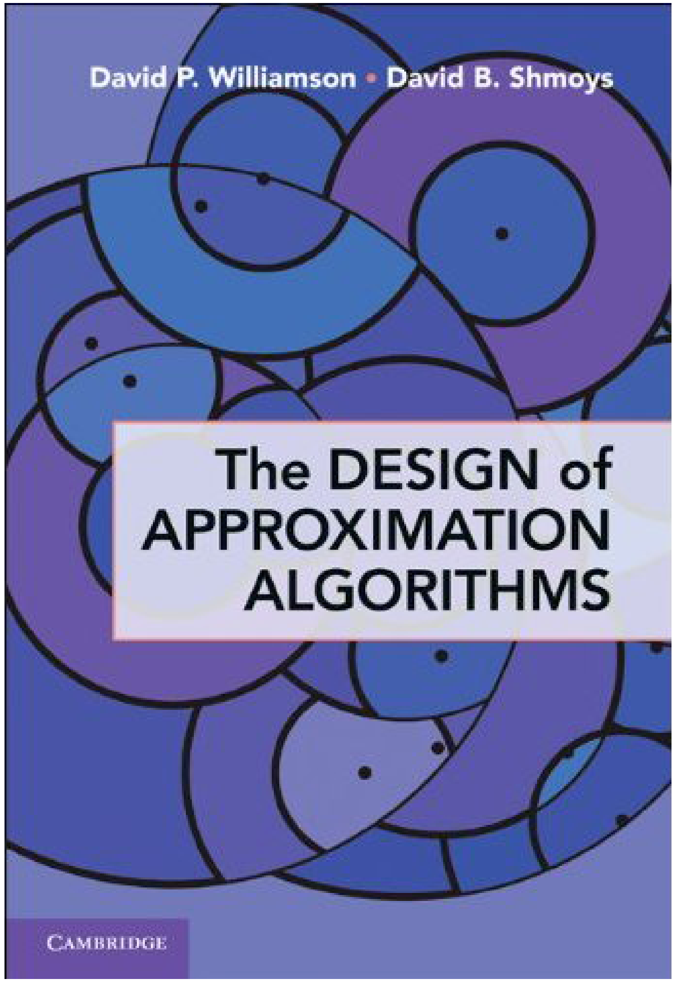


Theorem. Vertex Cover for bipartite graphs can be solved in $O(\sqrt{V}E)$ time.

Books



Vijay V. Vazirani
Approximation Algorithms
Springer-Verlag 2003



D. P. Williamson & D. B. Shmoys
The Design of Approximation Algorithms
Cambridge 2011

Approaches to NP-Hard Problems

- Exponential-time algorithms, e.g., backtracking
- Approximation algorithms:
  trade off: quality against running time
- Heuristics: experiments on benchmarks
- Randomization: find a needle in the haystack
- Design of parameterized algorithms

NEW

An Exact Algorithm for k-VC

BruteForceVC(Graph G, Integer k)

```
foreach C ∈ (V choose k) do
    // test whether C is a VC
    vc = true
    foreach uv ∈ E do
        if {u, v} ∩ C = ∅ then
            vc = false
    if vc then
        return ("yes", C)
return ("no", ∅)
```

$\left|\binom{V}{k}\right| = \binom{|V|}{k} = \binom{n}{k} = O(n^k)$

$O(E) = O(m)$

Runtime. $O(n^k m)$ – This is *not* polynomial in the size of the input ($= n + m$) as k is not a constant, but part of the input.

New Goal

Remark.
The class $\mathcal{FPT}$ does not change if $+$ is replaced with $\cdot$

Find an algorithm for k-VC with runtime

$$O(f(k) + |I|^c),$$

where $f \colon \mathbb{N} \to \mathbb{N}$ is a computable function (independent of I),
I is the given instance, c constant (independent of I)

That is, the runtime should depend
– arbitrarily on k, *degree of difficulty of the problem*
– polynomially in the size $|I|$ of instance I.

A problem that can be solved within this time bound is called *fixed-parameter tractable* with respect to the parameter k.

$\mathcal{FPT}$ = class of the fixed-parameter tractable problems.

Some Simple Observations...

Let $G = (V, E)$ be a graph and let C be a VC for G.
Suppose $v \notin C$ – which nodes are then certainly in C?

Obs. 1. If G is a graph, C a VC for G, and v a node, then: $v \in C$ or $N(v) \subseteq C$.

Consider the decision problem k-VC.
What holds for nodes of degree $> k$?

Obs. 2. Every node of degree $> k$ is contained in every k-VC.

What holds if $|E| > k^2$ and all nodes have degree $\leq k$?

Obs. 3. If $|E| > k^2$ and $\Delta(G) := \max_{v \in V} \deg v \leq k$, then G has no k-VC.

Algorithm of Buss

BussVC(Graph G, Integer k)

I) reduction to the kernel of the problem

$$\left.\begin{array}{l} C = \{v \in V \mid \deg v > k\} \\ \textbf{if } |C| > k \textbf{ then return } (\text{“no”}, \emptyset) \\ G' = (V', E') := G[V \setminus C] \text{ (without isolated nodes)} \\ k' = k - |C| \\ \textbf{if } |E'| > k^2 \textbf{ then return } (\text{“no”}, \emptyset) \end{array}\right\} \begin{array}{c} O(n+m) \\ \text{time} \end{array}$$

II) solution of the problem by brute force

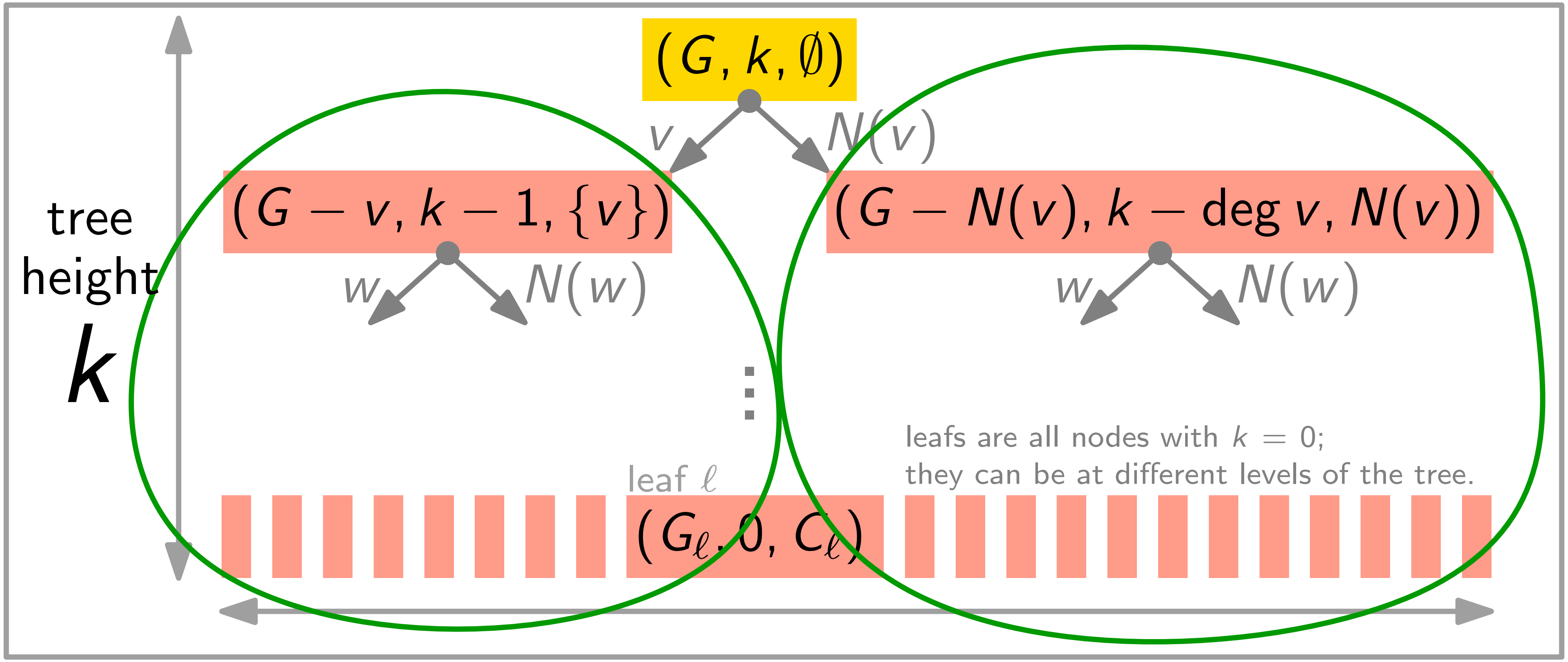
$$\left.\begin{array}{l} (\text{yesorno}, C') = \text{BruteForceVC}(G', k') \\ \textbf{return } (\text{yesorno}, C \cup C') \end{array}\right\} \begin{array}{l} O(m' \cdot (n')^{k'}) \text{ time} \\ \text{where } m' := |E'| \le k^2 \\ \quad \Rightarrow \; n' := |V'| \le 2k^2 \end{array}$$

Runtime. $O\big(n + m + k^2 \cdot (2k^2)^k\big) = O\big(\underbrace{n + m}_{|I|^1} + \underbrace{k^2 2^k \boxed{k^{2k}}}_{f(k)}\big)$

Thus: $k\text{-VC} \in \mathcal{FPT}$!

Search-Tree Algorithm

Idea. Improve phase II by building a search tree.



#nodes: $T(k) \leq 2T(k-1) + 1$, $T(0) = 1$ $\quad \Rightarrow T(k) \leq 2^{k+1} - 1 \in O(2^k)$

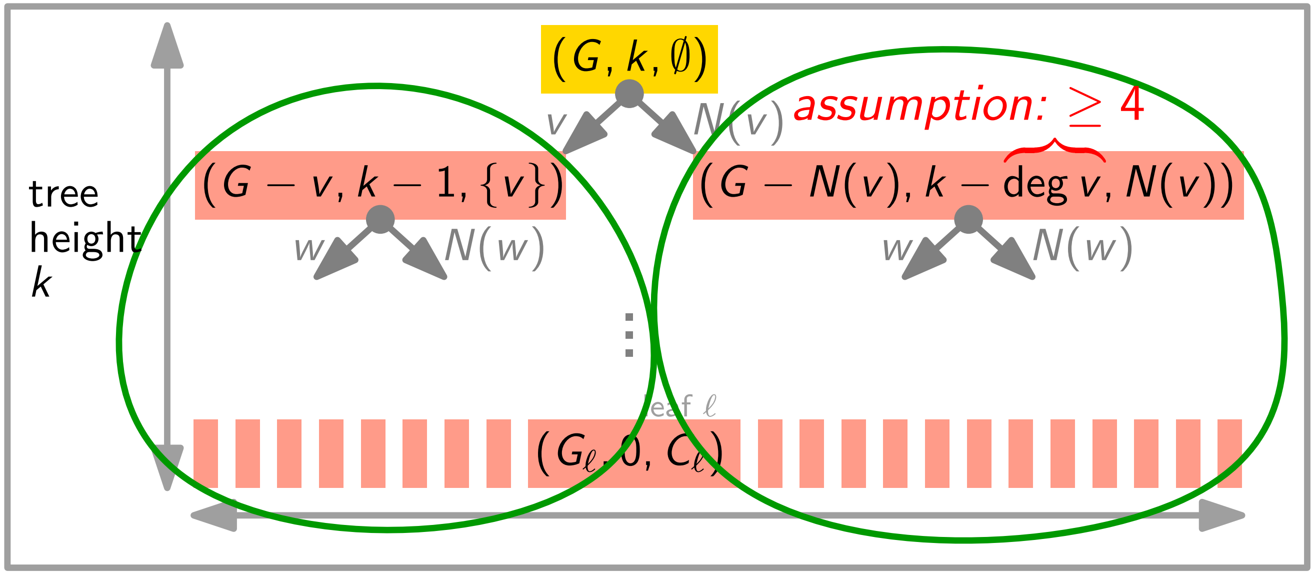
$\Rightarrow$ **Runtime:** $O^\star(2^k)$

If there is a leaf ℓ with $E_\ell = \emptyset$, then C_ℓ is a k-VC of G.

If there is no such leaf, then G has no k-VC.

The Degree-4 Algorithm

Idea. Improve estimation of $|N(v)|$.



$\Rightarrow$ $T(k) = T(k - \boxed{4}) + T(k - \boxed{1}) + 1, \quad T(\leq 4) = const.$

branching vector $(4, 1)$

Test $T(k) = z^k - 1 \quad \Rightarrow \quad z^k = z^{k-4} + z^{k-1}$ $\quad \cdot \frac{1}{z^{k-4}}$

$\Rightarrow$ characteristic polynomial: $z^4 = 1 + z^3$

$\Rightarrow$ largest positive solution: $z \approx 1.38$ (*branching number*)

$\Rightarrow$ $T(k) \in O(1.38^k)$. But how do we ensure $\deg v \geq 4$?

Kernels II

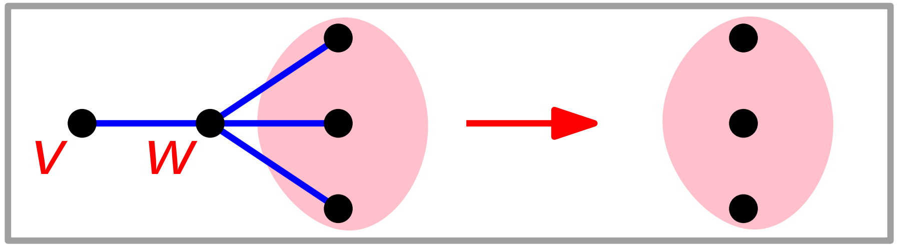
Before:

Rule K: eliminate nodes of degree $> k$

Regel 0: eliminate nodes of degree 0

improved kernels:

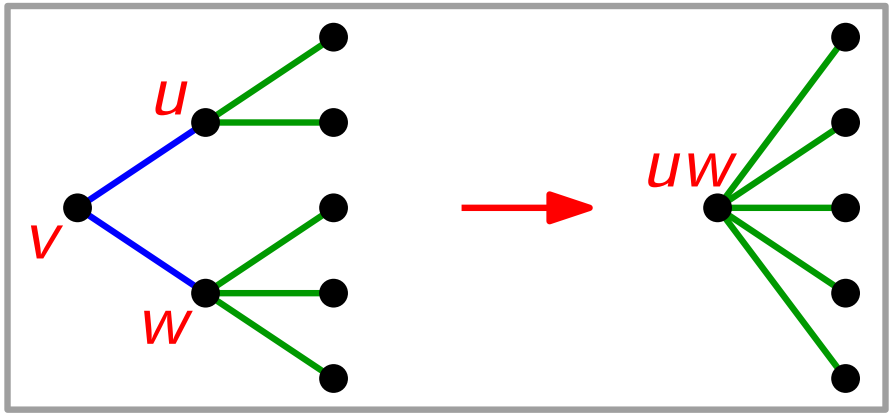
Rule 1: eliminate nodes of degree 1



$C = C' \cup \{w\}$

$k' = k - 1$

Rule 2: eliminate nodes of degree 2

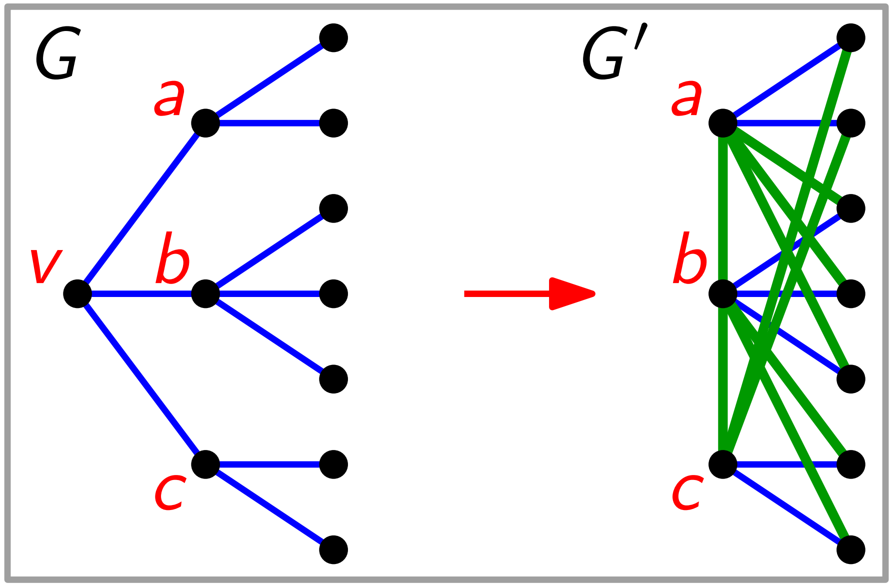


If $uw \in C'$,
take u and w in C,
otherwise v.

$k' = k - 1$

Rule 3: eliminate nodes of degree 3

Rule 3.1: $G[N(v)]$ contains no edge.



Claim. There is a k-VC in G $\Leftrightarrow$ there is a k-VC in G'.

...

Rule 3.2: There are edges in $G[N(v)]$.

...

The Degree-4 Algorithm

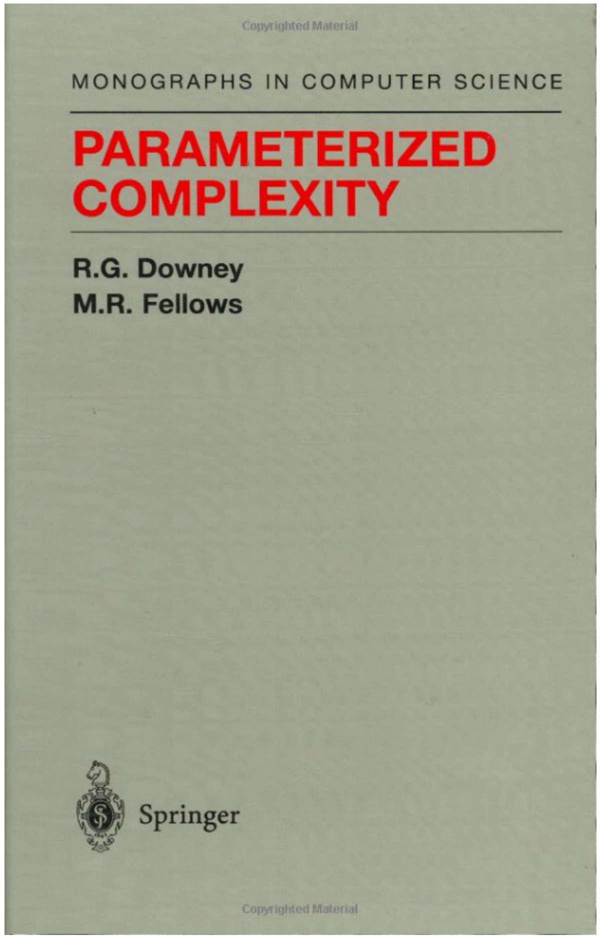
Idea: Apply the improved kernel rules to *every* node of the search tree *exhaustively*!

$\Rightarrow$ **Runtime**: $O(nk + k^2 \cdot 1.38^k) \quad \subseteq \quad O^\star(1.38^k)$

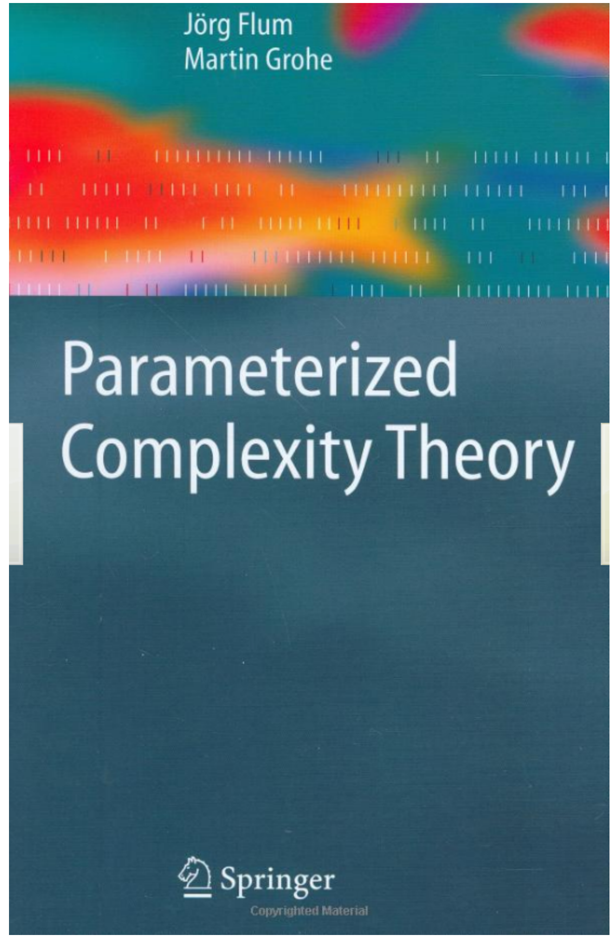
Conclusion

- k-VC can be solved in $O(nk + 1.38^k k^2)$ time.
- Parameterized complexity $=$
  new toolbox for NP-hard problems:
  kernels, tables, search trees, ...
- It is always useful to identify restricted parameters –
  FPT uses them!
- Hope:
  "natural" problem $P \in \mathcal{FPT} \;\Rightarrow\; f(k)$ reasonable.

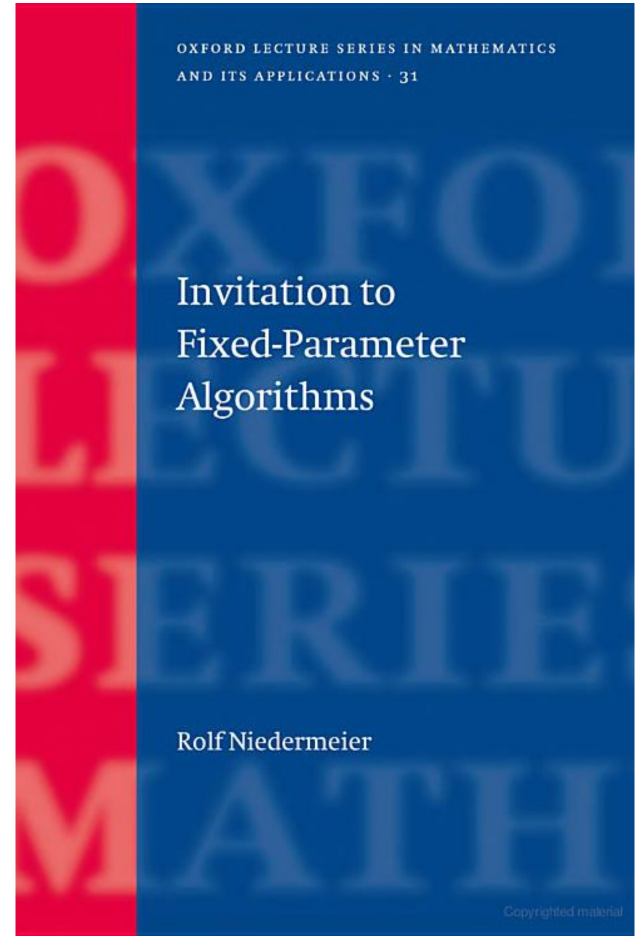
Books on FPT



1999



2006



2006